RFC 700
NIC 31020

A Protocol Experiment

Eric R. Mader
William W. Plummer
Raymond S. Tomlinson

I. Introduction

In early February, 1974 the main line printer on BBN's TENEX system failed and it was decided to use the PDP-11 line printer via the ARPANET both for the direct purpose of obtaining listings and also the indirect purpose of studying network protocols.

II. The Basic Protocol

The design was based on the protocol described by Cerf and Kahn in INWG Note #39. Familiarity with that document is assumed. The following is a brief sketch of the protocol. Not all features described in this section have been implemented. See Section VI.

At any instant, the sender has two pointers into the stream of bytes to be sent. Bytes to the left of the LEFT pointer have already been sent and acknowledged. Bytes in the "window" between the LEFT and RIGHT pointers have been sent (zero or more times), but no indication of successful transmission has been received. Bytes to the right of RIGHT remain to be considered at some time in the future.

In operation the sender is constantly sending bytes from the input data stream resulting in the RIGHT pointer advancing. Positive acknowledgements produced by the receiver cause the LEFT edge of the window to move towards the RIGHT edge.

LEFT and RIGHT are actually numerical byte positions within the data stream. The low order 16 bits of RIGHT are sent with each message as a sequence number so that the receiver can identify which part of the data stream it is receiving in case messages are not received in the same order they were transmitted. The receiver has a finite amount of buffer space available in which it can reassemble an image of the data in the transmitter's window. The receiver discards any messages which have sequence numbers outside of its buffer area. However, messages to the left of LEFT must be acknowledged even though they are discarded. Otherwise, a lost ACK would cause the sender to retransmit (and the receiver ingore) the message indefinitely. Messages received with bad checksums are also discarded.

As "good" messages are received, the holes are filled in the receiver's buffer and continuous segments at the left edge are passed to the physical line printer (in our case). The receiver informs the sender of

this action by sending an ACK (acknowledgement) message. This message specifies the sequence number of the byte it would like to receive next (the new value of LEFT in the sender) and the current amount of buffer space it has available (new maximum window width in the sender). The sender ignores ACK's to the left of LEFT and to the right of RIGHT. Thus, both the sender and receiver are prepared to handle multiple copies of messages.

Failures such as messages with bad checksums, messages lost during transmission (data and ACK's), and messages discarded due to sequences numbers which were apparently out of range, all manifest themselves to the sender as a dropped ACK. A dropped ACK will cause the sender's LEFT edge to stop advancing, leaving the unacknowledged message at the left of the sender's window, and possibly a corresponding hole at the left of the receiver's image of the window. Eventually, transmission will cease and a (10 second) timeout will trigger in the sender, causing retransmission of all data within the window. Note that at the instant of a timeout, there is no guarantee that the un-ACK'd message will be exactly at the left edge of the window or that it is the only unacknowledged message in the window. Retransmissions are likely to cause the receiver to see data that it has seen before, but duplicate messages will be discarded due to sequence number considerations.

III. "Say Again"

An extension to the INWN #39 protocol which was implemented was the ability to let the receiver force retransmission of the entire window by turning on a flag in any message back to the sender. This is useful in cases where the receiver believes that a data message has been dropped and it wants to force retransmission rather than wait for a timeout in the sender. Clearly, this relies on the network to preserve ordering of the messages. Also, it is not useful if the error rate is high because the whole window is retransmitted in order to get retransmission of a single message or two.

IV. Establishing an Association

In the experiment two flags were used to establish an association. FRST (FiRST flag) was the equivalent of SYN described in INWG Note #39 and served to identify the first message of an association. This instructed the receiver to accept the sequence number in the message as a definition of the starting point of sequence numbers for the association.

The second flag is a receiver-to-sender flag called HUH which is a request by the receiver for a definition of the sequence numbers. Upon receipt of a message containing an HUH, the sender responds by turning on FRST in the next data message. Normally, HUH is sent only if the receiver nad been restarted, or if it is replying to messages on a port

that it knows is not part of an association.

V. A Problem

A severe problem uncovered with the protocol was concerned with establishing an association. If the PDP-11 (receiver) was reloaded while the spooler (sender) was running, the first few pages of the data stream were printed about six times before normal operation was established. The cause was traced to the following sequence of actions:

1. The sender would be in a loop, timing out and retransmitting because the receiver had not responded.

2. Upon being restarted, the receiver would see a whole window's worth of messages, and respond to each with an HUH.

3. For each HUH the sender would reset the window and include a FRST flag with the first message in each of the (six) retransmissions.

4. The receiver would see the first message of the first retransmission containing a FRST, accept the sequence number, and print the data from that and the following messages. Then, another message containing the FRST flag would appear and the cycle would repeat (five more times). Note that the ACK's generated in the repetitions were ignored by the sender because they were to the left of the window.

As a "cure" for the above the receiver program was modified so that after sending an HUH, messages are ignored until one with a FRST flag appears. This solution is unacceptable in general because it leaves the receiver port useless if either the message containing the HUH or the response gets lost in transmission. Although a timeout was used to guard against this, the timeout cannot be trusted because it might cause two messages with FRST flags to be received -- just the problem which is being avoided!

An alternate cure which does not depend on the network to be lossless would be to modify the sender to respond to a HUH by ignoring all messages for at least a round trip delay time before sending its response containing the FRST flag. This results in having to define what this time is. In general this cannot be done when messages can become trapped for indefinite amounts of time in network partitions. This will be discussed more fully in a subsequent document.

VI. Features not Investigated

None of the programs to date have supported any of the following features:

1. Window size control. The window size was a constant (2048 bytes). In a future experiment the window size will be varied not only by indications of buffer space in the receiver, but also as a function of estimated transit time. (see below).

2. Reassembly. Since reassembly is conceptually easy, it is likely to be one of the first extensions. A message corrupter will be included in the receiver to test the functioning of the reassembly mechanism.

3. Expanded Internetwork Addresses

4. Multiple Associations

5. Reliable Making and Breaking of Associations

VII. Implementations Notes

The sender involves approximately ten pages of assembly code for the network message interface. Two processes are involved: one which fills a buffer by reading the input data stream, and a second process which sends network messages from the buffer and processes replies from the receiver. The two processes are joined by a coroutine mechanism, but in the future will be two parallel TENEX processes.

The receiver program consists of approximately four pages of BCPL code in addition to IO device drivers and routines which implement queueing primitives.

Each message contained between zero and 255 bytes of data arranged (as a coding convenience) in a way which is directly compatible with the BCPL string handling routines. Messages contained a single byte of checksum which was the low eight bits of the twos complement negation of the twos complement sum of all other bytes in the message. We recommend that some more reliable checksum function be employed in the future; even using eight-bit ones complement arithmetic would be better.

Source files for the various programs are available from the authors at Bolt Beranek and Newman, 50 Moulton Street, Cambridge Mass., 02138.

VIII. Simple Rate Calculations

If we assume that an active association has reached steady state, that processing delays are lumped into the transit time T, and that there are no errors, then the maximum data rate may be calculated as follows.

Assume the sequence numbers being passed by the RIGHT pointer are some function of time, R(t). Messages received by the receiver will be the same function of time but delayed T (a transit time) seconds. Since processing time is zero, the acknowledgments will bear this same function, R(t-T). Acknowlegements received by the sender will have sequence numbers R(t-2T).

Acknowledgements at the sender determine the LEFT pointer, L(t). Also, it is known that R(t) is ahead of L(t) by the width of the window which is a constant in steady state. Thus, we have the two relations:

$$L(t) = R(t-2T)$$
$$L(t) = R(t) - W$$

Now, let R(t) = Bt, i.e., sequence numbers are increasing linearly with time. (Microscopically, short bursts will alternate with longer periods of inactivity, but the average bandwidth will be B.) The result under the assumptions is that the bandwidth is:

$$B = W/2T \ .$$

That is, the bandwidth in bytes per second is just the steady state window width divided by the round trip delay time. Conversely, the aboe relation can be determine the buffer sized needed: in oreder for thee receiver to guarantee to accept information that was transmitted, it must supply buffering equal to (or greater than) the window size. The window size must be equal to or greater than the desired bandwidth times the round-trip delay time, i.e. equal to the number of messages in a round-trip "pipeline".

The bandwidth in the presence of a relatively low error rate may be calculated. Assume that B and W are expressed in terms of (full) messages rather than byte numbers. Each error has two effects: a time out delay of D seconds and retransmission of W messages. So, the time Q(M,N) required to transmit M messages burdened by N errors is the sum of the time to transmit the data once, N*D seconds of time out delay, and the time to transmit the window N more times.

$$Q(M,N) = (2T/W)*M + N*D + N*2T$$

Dividing by M to get time per message and muptiplying the last term by (W/W):

$$Q(M,N)/M = (2T/W) + (N/M)*D + (2T/W)*(N/M)*W \ .$$

But (M/N) is just the fraction of messages in error. Call this E.

$$Q(E) = (2T/W)*(1 + EW) + ED$$

$$B(E) = 1/[(2T/W)(1+EW) + ED]$$

The advantage to using the "say again" mechanism (Section III.) can now be seen: it forces D to be zero, allowing a reasonable average data rate in the presence of errors. Note the effect of a 10 second time out on a network with an E of 0.01 assuming W to be 20 messages and T of 0.5 second. B(D=10) is 6.7, but with forced retransmission, B(D=0) is 20.

IX. A Sequence Number Consideration

In order to reject duplicate messages, sequence numbers must contain a sufficient number of bits such that it is impossible to cycle through more than half the sequence number space in a message lifetime at maximum transmission rate. Assuming a 1 MegaByte per second network and a maximum lifetime of 500 seconds, the sequence number field of each message must be capable of holding the number 2*500*10**6 which is 10**9 or about 2**30. Thus, a 32-bit (4-byte) sequence number field is recommended.

X. Additional Control Functions

In response to an attempt to establish an association (SYN) it is felt that the receiver should be able to deny the attempt (RELease) in one of the following three ways:

REJECT. (I'm busy. Try again later.)
ABORT. (I don't understand what you are sending.
(Bad port, etc.))
ABNORMAL (SYN arrived on a established connection.)
(Receiver breaks connection and issues this REL.)

During an established association, the sender should be able to RELease the association in either of these ways:

DONE. (I'm done sending to you.)
GAG. (Stop. You are sending garbage (ACK's).)

These may be coded as combinations of bits in the FLAGS which are convenient for programming.